Comparative Reasoning for Knowledge Graph Fact Checking

Lihui Liu*, Houxiang Ji*, Jiejun Xu† and Hanghang Tong*
*Department of Computer Science, University of Illinois at Urbana Champaign
†HRL Laboratories, LLC.
Email: *lihuil2@illinois.edu, *hj14@illinois.edu, †jxu@hrl.com, *htong@illinois.edu

***Abstract*—Knowledge graph has been widely used in fact checking, owing to its capability to provide crucial background knowledge to help verify claims. Traditional fact checking works mainly focus on analyzing a single claim but have largely ignored analysis on the semantic consistency of pair-wise claims, despite its key importance in the real-world applications, e.g., multimodal fake news detection. This paper proposes a graph neural network based model INSPECTOR for pair-wise fact checking. Given a pair of claims, INSPECTOR aims to detect the potential semantic inconsistency of the input claims. The main idea of INSPECTOR is to use a graph attention neural network to learn a graph embedding for each claim in the pair, then use a tensor neural network to classify this pair of claims as consistent vs. inconsistent. The experiment results show that our algorithm outperforms state-of-the-art methods, with a higher accuracy and a lower variance.**

***Index Terms*—fact checking, knowledge graph, comparative reasoning**

I. INTRODUCTION

The knowledge graph is ubiquitous, and it provides prior knowledge and valuable information for many applications, e.g., knowledge graph alignment [1], question answering [2] and fact checking [3]. The goal of knowledge graph fact checking is to provide a more accurate, unbiased analysis of claims and rate them as consistent or inconsistent w.r.t. the background knowledge from the knowledge graph.

Although researchers have made great achievements in the fact checking area, e.g., [4] [5] [6], it is still highly challenging to check the semantic inconsistency among multiple pieces of information. The vast majority of the existing works primarily focus on checking the truthfulness of a single claim/triple, e.g., determining whether (`Alan Turing`, `wasBornIn`, `Canada`) is consistent or inconsistent. Very few of them focus on pair-wise fact checking, which aims to check the semantic (in)consistency of a pair of claims collectively.

However, in many real-world situations, e.g., multimodal fake news detection [7], single claim fact checking alone is insufficient. When we verify the two claims/triples at the same time, the result may be inconsistent even though each claim/triple is consistent if we evaluate it individually. For example, considering two claims/triples: (`Barack Obama`, `graduatedFrom`, `Harvard University`) and (`Barack Obama`, `majorIn`, `Political Science`). If we look at each of them individually, both of them might be true. But if we check them together at the same time, we can see that they cannot be both true, and thus the semantic inconsistency between them is spotted. This is because `Barack Obama` majored in `law` instead of `Political Science` when he studied at `Harvard University`.

TABLE I
NOTATIONS AND DEFINITIONS

Symbols	Definition
$\mathcal{G} = (V, R, E)$	the knowledge graph
v_i	the i^{th} entity/node in knowledge graph
r_i	the i^{th} relation/edge in knowledge graph
e_i	the i^{th} given by the user
G_i	the knowledge segment of e_i

In this paper, we aim to use the information in the knowledge graph to determine whether the claims/triples in the pair are consistent with each other. We propose a graph neural network model INSPECTOR to tackle the pair-wise fact checking problem. INSPECTOR finds an evidence pattern (a subgraph) for each claim to express the semantic meaning of it and utilizes a graph attention neural network to learn a graph embedding for each evidence pattern. A tensor neural network is then used to check the consistency of the pair of claims based on the learned graph embedding.

We summarize our contributions as follows

- **Pair-wise Fact Checking Model.** We propose a graph neural network based pair-wise fact checking model named INSPECTOR.
- **Empirical Evaluations.** We perform empirical evaluations to demonstrate the efficacy of INSPECTOR .

The rest of the paper is organized as follows. Section II introduces notations used in this paper and gives the problem definition. Section III introduces the overall framework of INSPECTOR and its technical details. The experiment results are presented in Section IV, and the related work is reviewed in Section V. Finally, the paper is concluded in Section VI.

II. PROBLEM DEFINITION

Table I gives the main notations used throughout this paper. A knowledge graph can be denoted as $\mathcal{G} = (V, R, E)$ where $V = \{v_1, v_2, ..., v_n\}$ is the set of nodes/entities, $R = \{r_1, r_2, ..., r_m\}$ is the set of relations and E is the set of triples. Each triple in the knowledge graph can be denoted as (h, r, t) where $h \in V$ is the head (i.e., subject)

978-1-7281-0858-2/19/$31.00 ©2022 IEEE

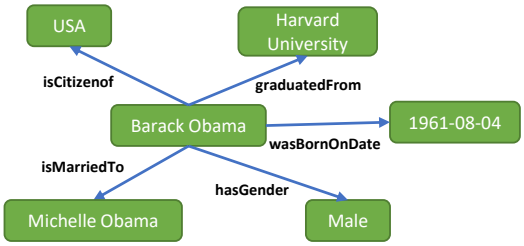

Fig. 1. An example of knowledge graph.

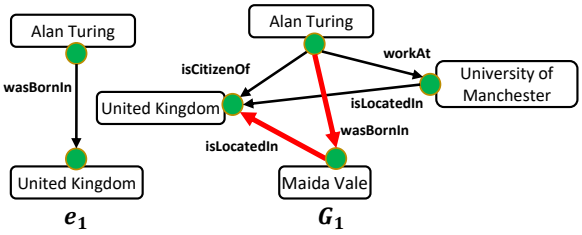

Fig. 2. Knowledge segment supporting the fact (Alan Turing, wasBornIn, United Kingdom). The red thick path shows the evidence.

of the triple, $t \in V$ is the tail (i.e., object) of the triple and $r \in R$ is the edge (i.e., relation, predicate) of the triple which connects the head h to the tail t. Figure 1 gives an example of the knowledge graph. In this graph, (`Barack Obama`, `isMarriedTo`, `Michelle Obama`) is a triple.

In this paper, we focus on claims which can be transformed into triples of knowledge graphs. Generally speaking, given a triple/claim from the user, we can find a *knowledge Segment* from the knowledge graph to express the meaning of this triple. Figure 2 gives an example of knowledge segment. Formally, a knowledge segment is defined as follows [3].

Definition 1: **Knowledge Segment** is a connected subgraph of the knowledge graph that describes the semantic context of a given triple.

Supposing we have two triples given by the user which are $e_1 =< \mathtt{s}_1, \mathtt{p}_1, \mathtt{o}_1 >$ and $e_2 =< \mathtt{s}_2, \mathtt{p}_2, \mathtt{o}_2 >$ ($\mathtt{s}_1, \mathtt{o}_1, \mathtt{s}_2, \mathtt{o}_2 \in V$ and $\mathtt{p}_1, \mathtt{p}_2 \in R$), and their corresponding knowledge segments are G_1 and G_2, respectively. The goal of INSPECTOR is to detect whether this pair of triples is consistent or not according to their knowledge segments.

Finally, the pair-wise fact checking problem can be formally defined as follows.

Problem Definition 1: pair-wise fact checking:

Given: (1) A knowledge graph $\mathcal{G}$, (2) two triples e_1 and e_2;

Output: A binary decision regarding the consistency of e_1 and e_2.

III. PROPOSED METHOD

In this section, we introduce the details of INSPECTOR. The overall framework of INSPECTOR contains four major steps, which are illustrated in Figure 3. First (**Knowledge Segment Extraction**), a knowledge segment is extracted from the knowledge graph for each claim/triple to express the semantic meaning of this claim/triple. Second (**Graph Level Embedding**), a cross graph attention neural network is used to embed each knowledge segment into a low dimension vector. Third (**Graph-Graph Interaction**), a neural tensor network is used to calculate the interaction (similarity) score of this graph embedding pair. Finally (**Consistency Prediction**), a fully connected neural network is used to predict the result. We highlight each step in the following four subsections.

A. Step 1: Knowledge Segment Extraction

Given a triple, the goal of knowledge segment extraction is to find a subgraph in the knowledge graph to express the semantic meaning of this triple [3]. Many algorithms have been proposed to extract the corresponding knowledge segment, e.g. multi-hop method [8], minimum cost maximum flow method [9], *K*-simple shortest paths based method [10] or connection subgraph extraction method [11], [12], etc.

In this paper, we apply an existing k-simple shortest path based subgraph extraction method [3] to find the knowledge segment. For the completeness of this paper, we briefly describe the main ideas of this method here. Given a triple, the knowledge graph will be first converted into a weighted graph by a TF-IDF like weighting strategy and then an approximate attributed subgraph matching method [13] is used to find the *k*-simple shortest paths [12] from the subject to the object of the given query triple as its knowledge segment.

B. Step 2: Graph Level Embedding

After finding the knowledge segment for each claim/triple, a cross attention neural network embeds each knowledge segment into a low dimension vector. Supposing we have a triple $e_1 =< \mathtt{s}_1, \mathtt{p}_1, \mathtt{o}_1 >$, and its knowledge segment is denoted as G_1. We use h_{ij}^t to denote the node representation of node j in e_i at t-th round of propagation, and x_{ij}^t to denote the node representation of node j in G_i at t-th round of propagation, h_{ij}^0 to denote the input feature vector of node j in e_i, x_{ij}^0 to denote the input feature vector of node j in G_i, and use h_{e_i}, x_{G_i} to denote the graph embedding of e_i and G_i, respectively. The cross attention neural network is defined as follows.

$$\begin{aligned} \mathbf{a}_{ij} &= \frac{\exp(h_{1i}^t \cdot x_{2j}^t)}{\sum_k \exp(h_{1i}^t \cdot x_{2k}^t)} \\ h_{1i}^{t+1} &= \text{MLP}(h_{1i}^t, \sum_j \mathbf{a}_{ij} * x_{2j}^t) \\ h_{e_1} &= \frac{1}{2}(h_{11}^T + h_{12}^T) \end{aligned} \tag{1}$$

where MLP is a multi-layer perceptron and $\cdot$ is the inner product. When calculating the graph embedding, we only propagate the information of G_1 to e_1, and do not update the node information in G_1. This is because, we treat G_1 as the background, and summarize its information into the graph embedding according to the information of e_1.

C. Step 3: Graph-Graph Interaction

Given the graph embedding of two knowledge segments produced by the cross attention neural network, multiple methods can be used to model the interaction, e.g., inner product, multi-layer perceptron, RNN, etc. In this paper, following the

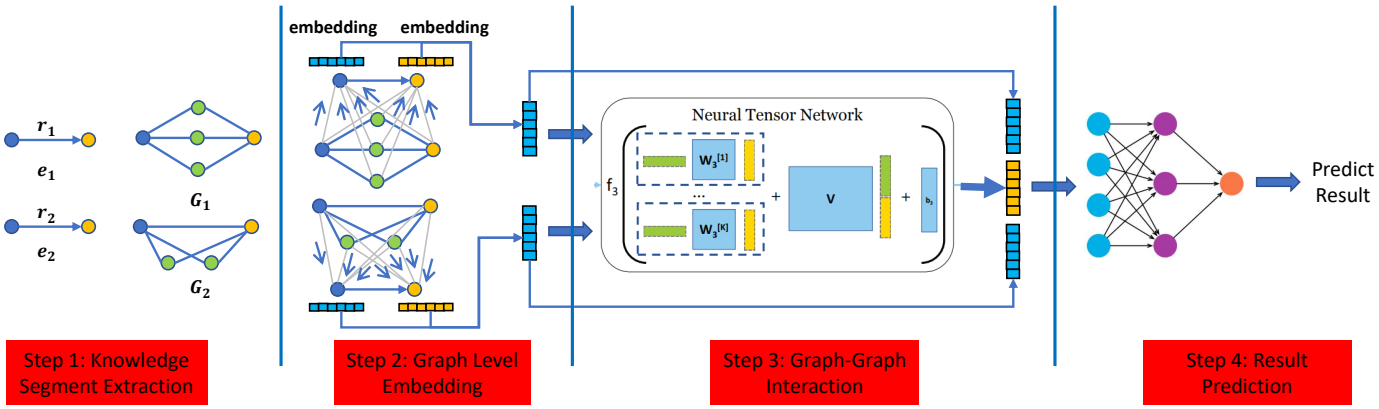

Fig. 3. INSPECTOR framework.

idea of [14], we use Neural Tensor Networks (NTN) to learn the interaction between two graph embeddings. The Neural Tensor Networks is defined as follows.

$$g(h_{e_i}, h_{e_j}) = f(h_{e_i}^T W_3^{[1:N]} h_{e_j} + V[h_{e_i}, h_{e_j}] + b_3) \quad (2)$$

where $W_3^{[1:N]} \in R^{D \times D \times N}$ is a weight tensor. [] is the concatenation operation, $V \in R^{N \times 2D}$ is a weight vector, $b_3 \in R^N$ is a bias vector, and $f()$ is an activation function. N is a hyperparameter. The output $g(h_{e_i}, h_{e_j})$ is a score vector with dimension $N \times 1$.

D. Step 4: Consistency Prediction

After obtaining the interaction scores, we concatenate the two graph embeddings and the interaction scores together and feed them into a fully connected neural network to predict whether they are consistent or not.

E. Training

During the training process, we generate a set of positive pairs and negative pairs, and minimize the loss function which is defined as follows.

$$L = \sum_{(i,j) \in \mathcal{D}} |\hat{s}_{ij} - s(e_i, e_j)| \quad (3)$$

where $\mathcal{D}$ is the set of training graph pairs, and $\hat{s}_{ij} = 1$ if (e_i, e_j) is a positive pair, otherwise $\hat{s}_{ij} = 0$. $s(e_i, e_j)$ is the result predicated by INSPECTOR.

F. Time Complexity

After getting the knowledge segment for each query triple, the time complexity of INSPECTOR contains two parts: Graph Level Embedding and Graph-Graph Interaction. The time complexity of cross attention network in Graph Level Embedding is $O(|V_{G_1}| + |V_{G_2}| + D^T)$ where $|V_{G_1}|$ and $|V_{G_2}|$ are the number of nodes in G_1 and G_2 respectively, D is the embedding dimension and T is the number of propagation round. The time complexity of neural tensor network in Graph-Graph Interaction is $O(D^2 N)$ where D is the embedding dimension and N is a hyperparameter given by the user.

IV. EXPERIMENTS

In this section, we conduct empirical studies to evaluate the performance of the proposed INSPECTOR. The knowledge graph we used is Yago [15] [1] which contains 12,430,705 triples, 4,295,825 entities, and 39 predicates. Five baseline methods are used, including two link prediction methods (TransE [6], Jaccard [16]), three fact checking methods (Knowledge Linking [17], KGMiner [18], Kompare [3]).

Ten query sets are used in the experiments, and each of them contains 300 query pairs. For each query set, we use 70%, 15%, 15% of them as training, test and validation data, respectively. For each positive query set, it contains positive query pairs that describe the true claims, while for each negative query set, it contains negative query pairs that describe the false claims. The dataset is from [3].

In the experiments, the embedding size for TransE is 64, and the training epoch is 1,200. For INSPECTOR, we use 2 propagation layers, and the embedding size is 64. We use the node embedding learned by TransE as the initial node embedding for INSPECTOR. All the experiments are conducted on a server with GPU Titan V 12GB.

A. Results

We use the accuracy as the evaluation metric in the experiments:

$$accuracy = \frac{C}{U} \quad (4)$$

where C is the number of queries correctly classified by the corresponding fact checking method and U is the total number of queries.

For the baseline methods (TransE, Jaccard, Knowledge Linking and KGMiner), when checking the consistency of a query pair $(\mathtt{s}_1, \mathtt{p}_1, \mathtt{o}_1)$ and $(\mathtt{s}_2, \mathtt{p}_2, \mathtt{o}_2)$, because these baseline methods were not originally designed for pair-wise fact checking, we use them to check the truthfulness of $(\mathtt{o}_1, \mathtt{isTypeOf}, \mathtt{o}_2)$ and $(\mathtt{o}_2, \mathtt{isTypeOf}, \mathtt{o}_1)$: if one of them is classified as true, this query pair is treated as true.

[1] It is publicly available at https://www.mpi-inf.mpg.de/de/departments/databases-and-information-systems/research/yago-naga/yago/downloads. We use the core version.

TABLE II
ACCURACY OF PAIR-WISE COMPARATIVE REASONING.

Dataset	# of queries	TransE	Jaccard	KL	KGMiner	Kompare	INSPECTOR
Family members positive	300	0.682	0.831	0.618	**0.983**	0.944	0.737
Family members negative	300	0.335	0.169	**1.000**	**1.000**	0.941	0.973
Graduated college positive	300	0.686	0.335	0.502	0.769	0.794	**0.974**
Graduated college negative	300	0.626	0.993	0.947	0.901	**0.994**	0.943
Live place positive	300	0.567	0.415	0.489	0.834	0.762	**0.951**
Live place negative	300	0.802	0.585	0.907	0.900	0.888	**0.976**
Birth place positive	300	0.590	0.435	0.537	0.698	0.800	**0.861**
Birth place negative	300	0.845	**1.000**	0.973	0.927	0.927	0.939
Work place positive	300	0.751	0.319	0.445	0.698	0.720	**0.902**
Work place negative	300	0.624	0.994	0.942	0.927	**0.995**	0.949
mean ± variance	-	0.651± 0.018	0.608± 0.092	0.736± 0.049	0.864± 0.011	0.877± 0.009	**0.921 ± 0.005**

Table II gives the details of the results. As we can see, most of the time, INSPECTOR can achieve the best results compared with the other baseline methods, and INSPECTOR has the highest average accuracy and the lowest variance.

V. RELATED WORK

In recent years, knowledge graphs have been widely used for fact checking. For example, in [19], the authors proposed a weighted knowledge stream/segment based method to check the truthfulness of a given claim. They used a TF-IDF strategy to transform the knowledge graph into a weighted graph, then made the predication according to the knowledge stream. In [19], they proposed a discriminative paths based method to predict the truthfulness of a given claim. When training the model, different training data are required for different predicates. Other methods, like TransE [6], Knowledge Linking [17] can also be used to check the truthfulness of a given claim. However,these methods are designed for single claim fact checking, and none of them focuses on pair-wise fact checking. In [3], the authors proposed an random walk graph kernel based method to check the truthfulness of two claims. They first used an influence function to measure the importance of each element in the knowledge segments, and then made the prediction according to these important elements.

VI. CONCLUSION

In this paper, we propose a pairwise fact checking algorithm called INSPECTOR. The goal of INSPECTOR is to detect whether there is semantic inconsistency between a pair of claims. The experimental results show that our algorithm has better performance compared with other methods on average, and these results illustrate the feasibility of neural networks on the emerging pair-wise fact checking area.

VII. ACKNOWLEDGEMENT

This work is supported by NSF (1947135, and 2134079), the NSF Program on Fairness in AI in collaboration with Amazon (1939725), DARPA (HR001121C0165), NIFA (2020-67021-32799), and ARO (W911NF2110088). The content of the information in this document does not necessarily reflect the position or the policy of the Government or Amazon, and no official endorsement should be inferred. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation here on.

REFERENCES

[1] Y. Yan, L. Liu, Y. Ban, B. Jing, and H. Tong, "Dynamic knowledge graph alignment," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 5, pp. 4564–4572, May 2021.

[2] L. Liu, B. Du, J. Xu, Y. Xia, and H. Tong, "Joint knowledge graph completion and question answering," in *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, ser. KDD '22. New York, NY, USA: Association for Computing Machinery, 2022, p. 1098–1108.

[3] L. Liu, B. Du, H. Ji, and H. Tong, "Kompare: A knowledge graph comparative reasoning system," 2021.

[4] H. Naeemul, A. Fatma, and C. Li, "Toward automated fact-checking: Detecting check-worthy factual claims by claimbuster," ser. KDD '17, 2017.

[5] W. Xiong, T. Hoang, and W. Wang, "Deeppath: A reinforcement learning method for knowledge graph reasoning," in *EMNLP*, 2017.

[6] B. Antoine, U. Nicolas, G. Alberto, W. Jason, and Y. Oksana, "Translating embeddings for modeling multi-relational data," in *NIPS '13*, ser. NIPS '13, pp. 2787–2795.

[7] K. Nakamura, S. Levy, and W. Y. Wang, "r/fakeddit: A new multimodal benchmark dataset for fine-grained fake news detection," *CoRR*, vol. abs/1911.03854, 2019. [Online]. Available: http://arxiv.org/abs/1911.03854

[8] C. Giovanni, S. Prashant, R. Luis, B. Johan, M. Filippo, and F. Alessandro, "Computational fact checking from knowledge networks," *PloS one*, vol. 10, 01 2015.

[9] P. Shiralkar, A. Flammini, F. Menczer, and G. L. Ciampaglia, "Finding streams in knowledge graphs to support fact checking," in *2017 IEEE International Conference on Data Mining (ICDM)*, 2017, pp. 859–864.

[10] S. Freitas, N. Cao, Y. Xia, D. H. P. Chau, and H. Tong, "Local partition in rich graphs," ser. BigData '19, Dec 2018, pp. 1001–1008.

[11] C. Faloutsos, K. McCurley, and A. Tomkins, "Fast discovery of connection subgraphs," in *KDD '04*. New York, NY, USA: ACM, 2004, pp. 118–127.

[12] Y. Koren, S. North, and C. Volinsky, "Measuring and extracting proximity in networks," ser. KDD '06. New York, NY, USA: ACM, 2006, pp. 245–255.

[13] L. Liu, B. Du, and H. Tong, "Gfinder: Approximate attributed subgraph matching," ser. BigData '19, Dec 2019.

[14] Y. Bai, H. Ding, S. Bian, T. Chen, Y. Sun, and W. Wang, "Simgnn: A neural network approach to fast graph similarity computation," in *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*, ser. WSDM '19, 2019, p. 384–392.

[15] F. M. Suchanek, G. Kasneci, and G. Weikum, "Yago: A core of semantic knowledge," ser. WWW '07. Association for Computing Machinery, 2007.

[16] D. Liben-Nowell and J. Kleinberg, "The link prediction problem for social networks," ser. CIKM '03.

[17] G. L. Ciampaglia, P. Shiralkar, and Rocha, "Computational fact checking from knowledge networks," 2015.

[18] B. Shi and T. Weninger, "Discriminative predicate path mining for fact checking in knowledge graphs."

[19] ——, "Discriminative predicate path mining for fact checking in knowledge graphs," *Know.-Based Syst.*, vol. 104, no. C, pp. 123–133, Jul. 2016.